UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/544,894	07/19/2006	Aravind R Dasu	117316-155055	6488

25943 7590 07/29/2010
Schwabe Williamson & Wyatt
PACWEST CENTER, SUITE 1900
1211 SW FIFTH AVENUE
PORTLAND, OR 97204

EXAMINER	
VU, TUAN A	

ART UNIT	PAPER NUMBER
2193	

MAIL DATE	DELIVERY MODE
07/29/2010	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	**Application No.** 10/544,894	**Applicant(s)** DASU ET AL.	
	Examiner TUAN A. VU	**Art Unit** 2193	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *7/19/06*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-62* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-12, 26-62* is/are rejected.

7)☒ Claim(s) *13-25* is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *8/5/05* is/are: a)☒ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date *See Continuation Sheet*.

4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application

6) ☐ Other: _____.

Continuation of Attachment(s) 3). Information Disclosure Statement(s) (PTO/SB/08), Paper No(s)/Mail Date :7/7/09(1); 7/7/09(2); 7/7/09(3).

DETAILED ACTION

1. This action is responsive to the application filed 7/19/2006.

Claims 1-62 have been submitted for examination.

Claims Objections

2. Claim 2 is objected to because of a mis-constructed or typed English phrase: the segment recited as "portion of the area *on which the block from which the block* has been partitioned" amounts to idiomatic or largely ill-constructed English; hence will be treated as 'portion of the area from which the block has been partitioned' in order to prosecute the merits.

3. Claim 2 is objected for the phrase "largest common subgraph common among a set of " also appears to be mis-typed or ill-constructed grouping of terms.

Claim Rejections - 35 USC § 112

4. The following is a quotation of the first paragraph of 35 U.S.C. 112:

> The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same and shall set forth the best mode contemplated by the inventor of carrying out his invention.

5. Claims 12-33 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

Specifically, claim 12 recites: *determining a relative delay among possible paths through the common subgraph*; performing branch and bound scheduling for the longest-delay-time path

and for less than all possible paths through the common subgraph including at least a longest-delay-time path

There insufficient description in the specifications for the mechanism of performing branch-and bound scheduling to "less than all possible paths through the common subgraph including at least a longest delay time path". The Appendix G describes sorting delay for worst delay and applying branch-and-bound scheduling whereas the scheduling algorithm applying to PCP and DFG extracted from a CDFG (Specs: pg. 43-45) has no mention of common subgraphs that include a longest-delay path, because nowhere this algorithm is tied to the section describing "largest common subgraph" or "common sub-expression" depicted elsewhere in the Specifications. The Disclosure discusses LSCG, and common subexpressions, PCP and branch-and-bound heuristic all in differing contexts. No where is there description as to how a 'relative delay' is determined *among possible paths through the common subgraph* (emphasis added). As claim 12 does not correlate DFG, common subgraph with relative delay and longest delay path, one would recognize that the inventor is not in possession of 'relative delay' and a heuristic using branch-and-bound that particularly address 'paths through the common subgraph' as claimed. It is not clear how the invention would teach one skill in the art for make use of branch-and-bound (or relative delay approach) scheduling as disclosed to make it applicable to 'common subexpression' or LCSG, as this would require undue experimentation. Therefore, the requirement recited as 'for less than all possible paths through the common subgraph' (for branch-and-bound and relative delay) will not be given patentable merit; and will be treated as 'for all possible paths including at least a longest/relative delay.

Claims 13-33 fail to cure to the lack of description deficiency of claim 12, hence are also rejected for non-enabling impropriety.

6. Claims 48, 55, 60 are rejected under 35 U.S.C. 112, first paragraph, as failing to comply with the written description requirement. The claim(s) contains subject matter which was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor(s), at the time the application was filed, had possession of the claimed invention.

The language recited (claims 48, 55, 60) as "providing for delay of a network schedule manager and of a logic schedule manager upon the processing element iteratively operating in excess of the scheduled operation" (*) is not deemed sufficiently described in the Specifications.

The portion of the Disclosure, written as the following paragraph, "*If P1's lifetime exceeds the assumed lifetime (most probable lifetime or a unit iteration), then all dependents of P1 and their dependents (both resource and data) should be notified and the respective Network Schedule Manager (NSM) and Logic Schedule Manager (LSM), of FIG. 27X, should be delayed. Of course, this implies that while preparing the schedule tables, 2 assumptions are made. [0333] 1) The lifetimes of solitary loops with unknown execution times are taken as per the most probable case obtained from prior trace file statistics (if available and applicable). Otherwise unitary iteration is considered. [0334] 2. All processes that are dependent on such solitary loop processes are scheduled with a small buffer at their start times. This is to provide time for notification through communication channels about any deviation from assumption 1 at run time*" appears to depict *lifetime* of a PI's exceeding some assumed value requiring adjustment via notification to NSM and LSM. Nowhere in this portion is shown 'providing for delay'

between NSM and LSM upon "the processing element iteratively operating in excess of the scheduled operation". As one skill in the art cannot make use of the claimed "providing" limitation (i.e. providing for delay of a NSM and a LSM), the above language (*) constitutes a lack of description, and to prosecute merits of the claim, the above would be treated as claim 47 regarding a management scheme with notification of a delay in additional processing elements, in view of the above paragraph.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. Claim 1 is rejected under 35 U.S.C. 103(a) as being unpatentable over Okada et al, USPN: 20010016936 (herein Okada) in view of Hammes et al, USPN: 7, 155,708 (herein Hammes) further in view of Brisk et al, "Instruction Generation and Regularity Extraction for Reconfigurable Processors", CASES 2002, Oct 2002, pp. 1-8 (herein Brisk) and Janssen et al, "A Specification Invariant Technique for Regularity Improvement between Flow-Graph Clusters", IEEE 1996, pg. 138-143 (herein Janssen)

As per claim 1, Okada discloses a method of fabricating a reconfigurable processor comprising:

producing, from a source code, a control data flow graph of data flow control flow and branch points (Fig. 7-8; Fig. 21; para 0036, pg.3);

extracting from the control flow graph code blocks (para 0007 - pg 1) lying between branch points (para 0036 pg.3; para 0105-0110; Fig.26-27);

generating intermediate data flow graphs from the code blocks lying between the branch points (Fig. 1, CDFG – Fig. 12, para 0091-0094 pg. 5 – Note: modifying of original CDFG with grouping reads on intermediate graph – see Fig. 12, Fig. 29);

identifying clusters shared among the intermediate data flow graphs at a highest level of granularity (para 0090, pg.5; Fig 7-8, Fig13);

determining, from the identified clusters, an unscheduled common subgraph shared among the intermediate data flow graphs (e.g. para 0166, 0174 pg. 10);

scheduling the unscheduled common subgraph; applying the scheduled common subgraph to the intermediate data flow graphs, including replacing the unscheduled common subgraph (para 0166, 0174, pg 10);

scheduling the intermediate data flow graphs containing the common subgraph for accomplishment of operations in the intermediate data flow graphs (Fig. 37, 38, 41) to derive data patches having operations and timings for individual intermediate data flow graphs (para 0106-110, pg. 6);

combining the data patches to combine operations and timing of the common subgraph with the operations and timings for individual ones of intermediate data flow graphs that are outside the common subgraph; scheduling for process time reduction from the combined data patches, multiple uses of the common subgraph operations and timings usable to accomplish operations and timings of all intermediate data flow graphs employing the common subgraph e.g. Fig. 12; para 0166-0176, pg. 10 – Note: use of selector to expand sub-CDFG logic scheduling to

other sub-CDFG to achieve output resolution of operations of nodes within the synthesis context of a integrated circuit reads on combining patches and timing from common subgraph to multiple uses of common subgraphs and timings to accomplish intermediate data flows – S14, S15, Fig. 12); and

implementing in hardware having mixed granularities the operations and timing of the largest common subgraph (S15, S16 – Fig. 12).

Okada does not explicitly disclose code blocks being extracted as basic blocks. The use of CDFG in terms of node being created as basic blocks is disclosed in **Hammes** (see Fig. 23-24; col. 7 lines 54-62); and based on the similarity by which input and output from nodes in Okada and Hammes are analyzed its most simplified form and submitted to a scheduler (see Okada: para 0090-0096; Hammes: Fig. 23-24) it would have been obvious for one of ordinary skill in the art to implement the pre-arranging of control flow graph in Okada so that these become basic block associated with data in/out operations as the context that they are sequentially analyzed along with scheduling of nodes with respect to their associated data dependency in its serial basic form as shown in Hammes (col. 21 line 63 to col. 22 line 21) because validation by a scheduler in regard to dependency of data associated to a basic block node without bifurcated I/O edges would be highly facilitated.

Okada does not disclose common subgraph as *largest common subgraph.* Okada discloses re-organizing CDFG into sub-CDFG based on operations analyzed at a high-level synthesis (para 0113, pg. 6; para 0126, pg.7; para 0103, pg. 5) and at the localized synthesis level provides selector to represent a common subgraph outputting the result from multiple inputs coming from upper subgraphs (para 0095, 0103, pg. 5; Fig 34a, 34b) where logical synthesis for

validating combinatorial logic among nodes within clusters being extended to nodes outside of clusters alleviate size of the whole circuitry (para 0131-0133; Fig 30-31) hence the reach out for the largest number of nodes under validation using incremental logical syntheses within a larger high-level synthesis scheme is disclosed. Similar to using control graph for synthesizing logical gates or CFG nodes and looking for common areas of operations shared by two upper subgraphs as in Okada (common multiplier - para 0166; common operator - 0174 pg. 10) **Brisk** discloses reconfigurable processors using CDFG in association with template algorithms to observe possibilities for cluster and validation of edge node requirements, where the creation of template information based upon the clustering phase allow scheduling heuristics (Brisk: sec 2 - pg. 2; sec 4 pg. 3) where every common slack pair graph (APCSG) is considered and sorted out for ASAP scheduling; that is, Brisk teaches that a large common slack values pre-determination (i.e. large scale extraction of regularity pattern) and associated clustering improves number of edges to implement (R col. bottom pg. 3) as parallel and sequential template-based APSCG heuristic approach enable merging and clustering. **Janssen**, similar to the flow-graph cluster endeavor as in Brisk, discloses 'regularity' improvement technique with a select operation (Janssen: sec 4 pg. 140) similar to the selector in Okada, pre-estimation between data path and flow graph, with emphasis on shared flow-graphs defined as share operation defined from multiple input ports (sec 4.1 pg. 140), wherein multiple operations can be represented by a select operation, and transformation based on shared Flow-graphs enable optimization of code (e.g. common subexpression elimination – sec 4.4 pg. 141) such that elementary transformation would be instrumental in making a composite transformation. Based on the overall effect to minimize code or logic gate size, and hardware circuitry via applying analysis of common operations, large

common slack, shared subgraph from above, it would have been obvious for one skill in the art at the time the invention was made to implement the scheduling of common subgraph in Okada at the localized gate synthesis level to the common subgraph is extended progressively to a largest common subgrah based on the improvement in hardware reduction and code size as taught, respectively, in Brisk, and Janssen. One would be motivated to do so because heuristics iterating the logical synthesis in Okada looking for replaceable candidate sets of common subgrahs (as taught in Janssen in light of Brisk) when expressed in terms of finding the most optimal amount of replaceable operation or node observed at a localized level prior to extending the logical analysis (i.e. local selector representation of multiple inputs) to more subgraphs outside a selected region, would be all the more effective since the largest number of common subgraphs obtained locally then extended globally as show in Janssen, would help reduce the size of the allocated memory for the gates (or software code) associated in implementing the circuit hardware, which is exactly the endeavor for using CDFG prior to generation of circuit as in Okada, and as in Brisk and Janssen.

9. Claims 12, 26-29, 32-33 are rejected under 35 U.S.C. 103(a) as being unpatentable over Okada et al, USPN: 20010016936 (herein Okada) in view of Hammes et al, USPN: 7, 155,708 (herein Hammes) further in view of Brisk et al, "Instruction Generation and Regularity Extraction for Reconfigurable Processors", CASES 2002, Oct 2002; and Janssen et al, "A Specification Invariant Technique for Regularity Improvement between Flow-Graph Clusters" further in view of Cheng et al, "High-Level synthesis: Current Status and Future Prospects", June 1993 pp. 351-400 (herein Cheng)

As per claim 12, Okada discloses a method of fabricating an integrated circuit implementing multiple program operations, comprising:

deriving control flow graphs for selected multiple program operations of a source code; identifying blocks of the control flow graphs; developing data flow graphs for a plurality two or more of the blocks (refer to claim 1);

identifying a common subgraph shared by at least a pair of the blocks of control (refer to claim 1) processes represented by the common subgraph; scheduling the shared processes for operation in individual ones of the multiple program operations (Fig. 12; para 0166-0176, pg. 10 – Note: use of selector to expand sub-CDFG logic scheduling to other sub-CDFG to achieve output resolution of operations of nodes within the synthesis context of a integrated circuit reads on scheduling i/o operations from blocks of control representing a common subgraph with respect to individual multiple operations construed via analyzed nodes and validated input/output data on the flow graph – S14, S15, Fig. 12);

scheduling of processing units to carry out the common subgraph (refer to claim 1), laying out an arrangement of circuit elements for implementation of the integrated circuit in hardware (Fig 12)

Okada does not explicitly disclose block of the control flow graphs as basic blocks; but this feature has been rendered obvious as set forth in claim 1.

Okada does not explicitly disclose *clustering the shared processes subgraph into a macroblock having nodes representing the shared processes and at least a plurality of unconditional, conditional and reconfiguration edges running between nodes.*

Okada discloses grouping (para 0135-0142) and scheduling based on selected path between branch operations involving unconditional/conditional edges and shared processes (Fig. 21-27; para 0166 - 0176) and forming a macro version as common block (sel - Fig. 28, 34) having internal node representing shared processes from upper nodes. The concept of shared process as subgraph or similar to a macroblock is disclosed in the regularity detection by Brisk with use of templates to support heuristics for parallel and sequential clustering (sec 3-4 pg. 2-5) and the resource sharing cluster by Janssen (sec 2-4 pg. 139-142) in which merging would capitalize the heuristics or combine elementary transformations (see Brisk: sec 4.1 pg. 5; see Janssen: composite transformation pg. 142). It would have been obvious for one of ordinary skill in the art to implement the grouping and scheduling of unconditional/condition edges between branch in Okada so that substitution of shared processes would be as a macroblock subgraph including nodes representing operations related to those unconditional/condition edges, such that reconfiguring would be based on those edges as taught in the template-based heuristics of Brisk or in view of shared-resource transformation in Janssen, because clustering of individual node operations representing intermediate edges (prior to branching) would enable resolution of data requirement while simplifying the layout of the complex flow graph elements; e.g. by replacing shared subexpression or common resource operations with a common macroblock; which facilitate allocation needed to implement the hardware physical layout (see Okada: Fig 12; para 0132-0133) pertinent to the ASIC development.

Nor does Okada explicitly disclose scheduling as including:

determining a relative delay among possible paths through the common subgraph; performing branch and bound scheduling for the longest-delay-time path and for less than all

possible paths through the common subgraph including at least a longest-delay-time path (Note: refer to USC 112 regarding weight not given to 'common subgraph' path teaching in relation to 'relative delay' and "branch-and-bound" scheduling)

The identification of delay among edges or path length was a known concern when deriving source code into CDFG and basic blocks from its DAG in the endeavor for scheduling and implementing hardware therefrom, as this is shown in Okada (operating time – para 0029-0030) and Hammes (Fig. 27), whereas applying of branch-and-bound heuristic is disclosed in Cheng as a technique used so to improve upon scheduling or sorting based on resources availability or register limitation derived along critical path or mobility (Cheng: pg. 363) or upon any phase related to resource allocation or look ahead heuristics (pg. 374-376), including variable binding or data transfer (pg. 386) and where delay-cost or area-cost can be complemented with branch-and-bound technology in the netlist synthesis approach (pg. 391, bottom); where scheduling procedures in terms of binding or data path allocation concerns with longest delay path that would not exceed a threshold (pg. 383) which is analogous to the concept of not exceeding a time in Okada (see Okada: para 0099, 0151). Based on the endeavor to localize delay and resources usage between node on the graph (e.g. DFG in Hammes) and in view of minimizing hardware placement area (surface size) in Okada's high-synthesis and path analysis with time threshold concerns according to Cheng proposals and Okada's path analysis, it would have been obvious for one of ordinary skill in the art to implement the scheduling in Okada so that delay is determined in conjunction of resources/register analysis respective to processes on the control graph (e.g. via basic block requirement between edges as in Hammes delay determination) with delay in terms of relative or longest delay paths, so that all *possible*

paths analyzed in heuristics about resource availability or input/output requirement, register limitation, variable binding, delay cost determination as set forth above would be enhanced by a branch-and-bound scheduling as taught in Cheng. One would be motivated to do so because this would fine tune the results afforded from the search regarding cost caused by possible path delay (relative or longest) set forth in Okada or Hammes, in view of the branch-and-bound approach positive benefits identified in Cheng from above.

Nor does Okada explicitly disclose merging all schedules including: *grouping the circuit elements into first level clusters; and placing the first level clusters by grouping the first level clusters together to form second level clusters and placing the second level clusters.*

The grouping of nodes in terms of cluster is disclosed in Brisk where combining results from either parallel/sequential clustering whereas transformation from elementary to composite is effectuated in Janssen's technique to identify shared resouces (see above). Based on the well-known concept in ASIC design with reconfigurable elements where placement of HW element is based on scheduling prior to allocation and layout, the grouping of first level cluster and second level clusters in view of the high-level synthesis would be deemed obvious. It would have been obvious for one of ordinary skill in the art to implement the grouping and analysis of data dependency related to shared processes as in Okada in view of both Brisk and Janssen, so that circuit elements are grouped based on such synthesis and clustering approach, including placing the first level clusters by grouping the first level clusters together to form second level clusters and placing the second level clusters, because this placement would be the ultimate phase for which detection of shared processes, resources or subgraphs has been extensively employed – as per Brisk, Janssen and even Okada -- in any stage prior to the allocation phase.

As per claim 26, Okada does not explicitly disclose wherein said scheduling the shared processes represented by the common subgraph includes comprises ASAP scheduling the common subgraph. Cheng discloses use of branch-and-bound to complement heuristics with identification of critical path, mobility of resources, all of which scheduling aspects considered integral to a preliminary algorithm known a constructive or iterative algorithm (sec 3.11.2 pg. 361 to 363) which also includes analyzing CFG and DFG with ASAP approaches for finding boundaries of resources. It would have been obvious for one of ordinary skill in the art to implement the resources finding and execution time related thereto in Okada (see claim 12) so that a ASAP approach would be applied because this would effectuate a non-exhaustive collection of data needed for the latter stage of scheduling to implement branch-and-bound heuristics for the reasons set forth in claim 12, in view of the proposed approaches laid out in Cheng

As per claim 27, Okada discloses ASIC as conventionally a field of design where high-level synthesis would be used to reduce size (para 0004, para 0041), hence it would have been obvious for one of ordinary skill in the art to implement Okada in providing the common operations of the common subgraph so that this providing would be for an application specific integrated circuit, because of the well-known applicability of high-level synthesis in reducing size of the surface to implement the circuit.

As per claim 28, Okada discloses does:

identifying at least one other common subgraph shared by the at least a pair of the basic blocks (refer to claim 1); but does not explicitly disclose:

(i) scheduling other shared processes represented by the other common subgraph; scheduling the other shared processes for operation in individual ones of the multiple program operations; and

(ii) laying out another arrangement of other circuit elements for implementation of the integrated circuit in hardware, including:

(iii) grouping the other circuit elements into other first level clusters; and

(iv) placing the other first level clusters by grouping the other first level clusters together to form other second level clusters and placing the other second level clusters.

But in view of the intent to systematically address all level of the CFG and the modified aspect of the modified subgraphs via scheduling and grouping, the above steps (i) to (ii) along with (iii) and (iv) would have been obvious, because any other shared process not addressed in the synthesis and hardware placing via the process included in the layered clustering or grouping as set forth in claim 12 would fail to achieve the size reducing contemplated in Okada's high-level synthesis and hardware allocation and chip building as purported.

As per claim 29, Okada does not explicitly disclose identifying a largest common subgraph shared by the at least a pair of the basic blocks; however, this limitation has been addressed in claim 1.

As per claim 32, refer to claim 27.

As per claim 33, refer to computer-readable medium (Okada: claim 8 - pg. 8) to perform the method of claim 12.

10. Claim 11 is rejected under 35 U.S.C. 103(a) as being unpatentable over Okada et al, USPN: 20010016936 (herein Okada) in view of Brisk et al, "Instruction Generation and

Regularity Extraction for Reconfigurable Processors", CASES 2002, Oct 2002, pp. 1-8 (herein Brisk) and Janssen et al, "A Specification Invariant Technique for Regularity Improvement between Flow-Graph Clusters", IEEE 1996, pg. 138-143 (herein Janssen)

As per claim 11, Okada discloses a method comprising:

providing source code for an application to be run by an application-specific reconfigurable circuit (ASIC – para 0004-0007, pg. 1);

deriving, from the source code, flow graphs representing separate portions of the application (refer to claim 1);

identifying at least one common flow graph from at least two of the separate portions of the application (refer to claim 1); and

configuring circuitry of the application-specific reconfigurable circuit to be shared by the separate portions of the application (Fig. 12; para 0166-0176, pg. 10 – Note: use of selector to expand sub-CDFG logic scheduling to other sub-CDFG to achieve output resolution of operations of nodes within the synthesis context of a integrated circuit **reads on** combining and configuring circuitry of the application-specific reconfigurable circuit – S14, S15, Fig. 12).

Okada does not teach common subgraph specifically as *largest common subgraph*; however this largest common subgraph limitation has been addressed in claim 1.

11. Claims 5-10 are rejected under 35 U.S.C. 103(a) as being unpatentable over Gutberlet et al, USPN: 7, 120,879 (herein Gutberlet) further in view of Cheng et al, "High-Level synthesis: Current Status and Future Prospects", June 1993 pp. 351-400 (herein Cheng)

As per claim 5, Gutberlet discloses a method of scheduling process elements of hardware implementing a program operation, comprising:

developing a control data flow graph from software (col. 9 line 35 to col. 10 line 21);

using a first and a second scheduling algorithm (one or more times … incremental heuristic algorithm – col. 11 lines 19-34) to arrive at a first and second scheduling of the process elements for at least one and less than all selected paths of the control data flow graph (Fig. 5, Fig. 6a-6d; *two stages*, col. 10 lines 25-47 – Note: receiving result from one schedule instance and expanding/collapsing- see col. 5 lines 5-25 - result by the designer reads on using first and second scheduling of elements of selected paths - see col. 11 lines 42-55).

Gutberlet does not explicitly disclose using first and second algorithm for all selected paths to reduce a time of execution thereof. The endeavor of Gutberlet's GUI gantt-chart approach as to present icons with size, shape or color - as a result of scheduling selected paths - entails support for the designer with the knowledge including no, minimal or actual delay within logic connections (col. 17 lines 12-57) for developer modification purposes. Using a CDFG and scheduler prior to allocate resources in actual design and build of hardware circuit was a known concept of high-level synthesis by which a developer can reduce power consumption and maximizing speed as this is disclosed in Cheng (see Cheng: *maximization of … speed, minimization of pin number, power consumption* - pg. 352, top para). It would have been obvious for one of ordinary skill in the art to implement the developer's interface in Gutberlet so that the modifiable and viewable scheduling result in the Gantt-based and multiple scheduler approach would include the purpose to reduce time of execution of paths being targeted for one or more scheduling stages, because of the well-known practice in using graph and scheduling prior to allocation of real-world resources to implementing circuit as endeavored in the integrated chip design/building shown by Cheng.

Nor does Gutberlet disclose first non-exhaustive scheduling algorithm and a second exhaustive or non-exhaustive scheduling algorithm for at least one and less than all selected paths of the control data flow graph and *merging* all schedules, according to data and resource dependencies. Gutberlet discloses a preliminary stage of scheduling without feedback and a successive stage with interactive feedback based on which to invoke the scheduler for further cycle of simulation (col. 10 lines 25-47) where the scheduler can call a ASAP in conjunction with other incremental heuristics (col. 11 lines 11-31), hence invoking a first stage non-exhaustive algorithm is recognized. Cheng discloses iterative/constructive algorithms for scheduling with ASAP or CADDY(Cheng: pg. 362 bottom; sec. 3.1.1.1 pg. 361) as in Gutberlet, where Cheng's iterative approach includes critical path scheduling for sorting ready operations according to mobility (pg. 363 top para) with concern of register resource prior to applying another scheduling. Based on Cheng use of global rules and allocation in the iterative approach (pg. 370 bottom) and allocation problem by which a *branch-and-bound* technique can be used to update a best result of a current search of another global allocation technique (Cheng: *branch-and-bound ... more efficient* - pg. 374, top; sec 3.1.1.1 pg. 361; second phase … branch-and-bound – pg. 386 middle), it would have been obvious for one of ordinary skill in the art to implement the two stage scheduling in Gutberlet so that a non-exhaustive stage is followed by a more exhaustive scheduling approach using the likes of *branch-and-bound* to update a early stage such as a critical path search of register allocation as taught above in Cheng, because more exhaustive stage following a preparing stage would yield the intended results as contemplated in scheduling using iterative approach by both Gutberlet and Cheng, i.e. using developer's input leading to minimizing of size and maximizing of speed as set forth above.

As per claims 6-7, Gutberlet does not explicitly disclose, wherein said using a first non-exhaustive scheduling algorithm step (b) comprises includes Partial Critical Path scheduling. wherein said using a second exhaustive or non-exhaustive scheduling algorithm step (c) comprises includes branch and bound based bound-based scheduling. Cheng discloses iterative approach with a fast stage to capture critical path with register allocation/mobility requirement and a more in-depth stage using branch-and-bound analysis (Cheng: *branch-and-bound ... more efficient* - pg. 374, top; sec 3.1.1.1 pg. 361; second phase … branch-and-bound – pg. 386 middle) to update upon a allocation result performed by a previous global technique among the iterative scheduling paradigm (refer to claim 5); such that the non-exhaustive algorithm (b) followed by more exhaustive algorithm (c) from above would be deemed obvious in view of the rationale set forth above in claim 5, for the same reasons.

As per claims 8-10, Gutberlet discloses VHDL and design of electronic circuit (col. 1), hence it would have been obvious for one of ordinary skill in the art to implement the HDL, the scheduling and allocation in Gutberlet (col. 8-10) so that these mechanisms support developing an-integrated circuit configured to perform the program operation scheduled according to, respectively claims 5, 6, and 7 in view of the rationale set forth therein and the ultimate purpose in using VHDL, CDFG, RTL, scheduling in chip design applications.

12. Claim 2 is rejected under 35 U.S.C. 103(a) as being unpatentable over Okada et al, USPN: 20010016936 (herein Okada) and further in view of Brisk et al, "Instruction Generation and Regularity Extraction for Reconfigurable Processors", and Janssen et al, "A Specification Invariant Technique for Regularity Improvement between Flow-Graph Clusters", further in view of APA(Admitted Prior Art: Specifications pg. 5-10)

As per claim 2, Okada discloses a method of making an integrated circuit comprising:

identifying hardware circuit elements (Fig. 16, 18, 19, 20, 28-29) for execution of a common subgraph (para 0166, 0174 pg. 10) [[common]] among a set of flow graphs representing a programmed operation (Fig. 3, 5, 14-15, 22-24, 33, 35, 41-42);

partitioning into blocks the hardware circuit elements (Fig. 12);

partitioning into sub-blocks the hardware circuit elements of individual blocks (Allocation, generation of circuit – Fig. 12); and

Okada does not teach common subgraph specifically as *largest common subgraph*; however this largest common subgraph limitation has been addressed in claim 1.

Nor does Okada explicitly disclose:

arranging the blocks on an area representative of an available area of a surface of a substrate on which the circuit elements are to be formed; routing interconnections among the blocks; arranging individual sub-blocks on a portion of the area on which the block from which the block has been partitioned, routing interconnections among the sub-blocks, and iteratively partitioning and routing among lesser sub-blocks until the hardware circuit elements have been placed and routed. Okada discloses grouping of CDF elements (e.g. para 0135-0153) using analytical synthesis of requirement between nodes or graph elements to move or replace them in an effort to reduce scale (para 0098, 0104, 0161, 0163) of area chosen to implement the corresponding hardware circuit elements. Analogous to the designing of integrated circuit of Okada, APA discloses chip design with partitioning of blocks into sub-blocks of the hardware elements in area of surface substrate(surface planning, recursively into smaller blocks – APA, pg. 9), with routing of interconnections and portioning, placement and routing (APA, Specs pg.

8) using iterative placement until size of chip is reduced (APA, pg. 9-10). Based on the well-known intent to improve allocation of resources to a preferably reduced area of consumption by HW elements(see APA, pg. 5 bottom) via scheduling and reconfiguring of graphs as in Okada, it would have been obvious for one of ordinary skill in the art to implement the chip or ASIC manufacturing in Okada's approach for generating a circuit so that the result from reconfiguring the CDFG and allocation of gate elements would be subjected to arranging/partitioning of blocks into a surface substrated on which circuit elements are to be formed, including routing interconnections and arranging into sub-subblock, the routing and partitioning being done iteratively as taught in APA until all the HW elements targeted in the build are included in a satisfactorily reduced size, because the endeavor for using synthesis and modification of subgraphs in Okada has been no other than to reduce size of the circuit (see Okada: reducing the circuit – para 0177, pg. 11) and via iterative placement and routing as by APA, this endeavor would stand a much better to be realized.

13. Claims 3-4 are rejected under 35 U.S.C. 103(a) as being unpatentable over Okada et al, USPN: 20010016936 (herein Okada) and further in view of Brisk et al, "Instruction Generation and Regularity Extraction for Reconfigurable Processors", and Janssen et al, "A Specification Invariant Technique for Regularity Improvement between Flow-Graph Clusters", further in view of APA(Admitted Prior Art: Specifications pg. 8-10) and Bertolet et al, USPN: 5,671,432 (herein Bertolet)

As per claims 3-4, Okada does not explicitly disclose that routing interconnections among the blocks includes comprise locating conductors and switches for interconnections among blocks, sub-blocks, and circuit elements, wherein said locating conductors and switches

includes locating variable switches to effect variable conductive paths among the blocks, sub-blocks and circuit elements. Similar to designing hardware and using of FPGA (see APA: pg. 3, 9) in support for applications to configure ASIC architecture as in Okada, **Bertolet** discloses conductor and switches among the integrated gates so to enable conduction and propagation with selective control passing between bus support of the interconnected programmable elements (col. 5 lines 4-23). It would have been obvious for one of ordinary skill in the art to implement the ASIC design in light of the interconnection phase and reconfigurable element of gates as taught in the purport of Okada and APA so that conductors and switches among blocks, subblocks and circuit elements were used to effect selection or variation of paths among the interconnected elements as taught in Bertolet, as this would enhance the iterative placement and reconfiguration endeavor adapted to effectuate the best optimized path transmission (i.e. via selective switching of transfer) leading to minimized consumption of power so well contemplated in the field of FPGA design and integrated circuit building.

14. Claims 30-31 are rejected under 35 U.S.C. 103(a) as being unpatentable over Okada et al, USPN: 20010016936 (herein Okada) in view of Hammes et al, USPN: 7, 155,708 (herein Hammes) further in view of Brisk et al, "Instruction Generation and Regularity Extraction for Reconfigurable Processors", CASES 2002, Oct 2002; and Janssen et al, "A Specification Invariant Technique for Regularity Improvement between Flow-Graph Clusters" and Cheng et al, "High-Level synthesis: Current Status and Future Prospects", June 1993 pp. 351-400 (herein Cheng); further in view of Bertolet et al, USPN: 5,671,432 (herein Bertolet)

As per claim 30-31, Okada does not explicitly disclose providing switching of differing delays among processes of the common subgraph to effect subgraphs operating each individual

ones of the selected_multiple program operations, including providing multiplexers operative to apply alternative delays between processes of the common subgraph. Okada discloses circuit implementation with multiple inputs and a selector (e.g. Fig. 9, 16, 28, 34) which is reminiscent of a MUX gate, a HW concept along with that of switches which were also mentioned in Cheng(MUXes – middle pg. 379; Muxes, Switches - Table 1 - pg. 354) whereas the concept of using a switch to transfer control among alternated data flow or bus related to other gates is disclosed in Bertolet (refer to claim 4) hence the rationale as to providing a switching or selector functionality among different subgraphs such as switches or multiplexers operative to apply alternative delays between processes of the common subgraph would have been obvious for the same reasons as set forth in claim 4.

15. Claims 34-37 are rejected under 35 U.S.C. 103(a) as being unpatentable over Cheng et al, “High-Level synthesis: Current Status and Future Prospects”, June 1993 pp. 351-400 (herein Cheng) in view of APA (Admitted Prior Art, Specs, pg. 1-10)

As per claim 34, Cheng discloses a method including

scheduling at least a portion of the operations by calculating a delay (Table 3, CADDY – data flow; *data-flow graph … expected delay* – sec 3.2 pg. 378) along individual paths through a data flow graph, wherein the paths go from a processing element being scheduled to a sink node of the data flow graph (*look ahead scheduling, CADDY, list scheduling* - pg. 364, top; *ordered … each operation .. providing the resources … ordered* – pag. 362 - Note: order of operations adopted for list or look-ahead scheduling where node observed in scheduling provides resources to the next operation in sequence reads on paths to a sink node – list scheduling, look ahead - table 2, pg. 372), wherein said scheduling includes:

scheduling as a longest in duration path, a plurality of longest of processing times of processing elements, including the delay and the reconfiguration delay (longest path, maintains data dependencies – pg. 383) including an effect of the reconfiguration on a delay time (SEHWA: pg. 383 - Note: algorithm for longest execution time watch over a limit using data flow and cost constraints and time limit as well as latency reads on including effect of delay timie)

Cheng does not explicitly disclose method for fabricating a reconfigurable integrated circuit and that reconfiguration is part of a integrated circuit. However, Cheng mentions that synthesis can be use in pipelined ASICs (pg. 383) while graphical interface and design tool can support *integrated circuits* designers. And in view of the integrated circuits development contemplated via use of Cheng's high-level synthesis and the field of design for building ASIC and reconfigurable solutions with FPGA as taught in APA (Specifications : pg. 2-3) for planning, placement and routing of physical elements (APA: pg. 9), it would have been obvious for one of ordinary skill in the art to implement Cheng synthesis approaches in light of APA to support fabrication of integrated circuits as this HW fabricating field does require reduction in physical layout or space size which the high-synthesis by Cheng can provide (see *reducing chip size* - sec 3.1.1 pg. 370; minimize chip area – pg. 378).

Nor does Cheng explicitly disclose scheduling and data-flow graph in terms of adding to edges of the data flow graph reconfiguration edges representing a reconfiguration of that part of circuit effecting the at least a portion of the operations. But the building of a data flow graph amounts to adding element to the flow graph based on reading or parsing corresponding software configuration or code and this is seen in Cheng (see Figure 1, pg. 356) in order for dependencies

of data to be incrementally expressed. Based on the purpose of develop circuit using VHDL, it would have been obvious for one of ordinary skill in the art to implement Cheng's scheduling for expected delay via a DFG so that the construction establishing data dependencies or latency (see CADDY, latency optimization – see pg. 364; Table 3, CADDY – data flow) would enable adding of edges representing a reconfiguration of that part of circuit effecting the at least a portion of the operations, because constructing a graph for thoroughly establishing needed allocation for the circuitry would fail if edges are not added constructively for supporting data flow information needed in the scheduling.

As per claim 35, Cheng discloses scheduling one or more_shorter- in-duration paths within a time established for the longest-in-duration path (Note: heuristics observing longest path by inherency would include detection of one or more relatively shorter paths with respect to the longest's).

As per claim 36, Cheng discloses a *integrated circuit* fabricated according to the method of claim 34 (refer to the rationale in claim 34)

As per claim 37, Cheng (in view of APA) discloses computer-readable medium comprising stored programming instructions for execution by a computing device, the instructions comprising:

instructions to schedule at least a portion of operations by calculating a delay along individual paths through a data flow graph, wherein the individual paths go from a processing element being scheduled to a sink node of the data flow graph;

instructions to add to edges of the data flow graph reconfiguration edges representing a reconfiguration of that part of the integrated circuit effecting the at least a portion of the operations; and

instructions to include an effect of the reconfiguration on a delay time; and

instructions to schedule, as a longest in duration path, a plurality of longest of processing times of processing elements, including the delay and the reconfiguration delay;

all of which having been addressed in claim 34.

16. Claims 38-43 are rejected under 35 U.S.C. 103(a) as being unpatentable over Cheng et al, “High-Level synthesis: Current Status and Future Prospects”, June 1993 pp. 351-400 (herein Cheng) in view of APA, further in view of Dangelo et al, USPN: 5,572,436 (herein Dangelo)

As per claim 38, Cheng discloses a method comprising

scheduling at least a portion of operations of an circuit including performing a first non-exhaustive calculation of a plurality of delays (refer to claim 34 – Note: first approach using look ahead or list scheduling in CADDY method -- reads on non-exhaustive initial search which is to be followed by more exhaustive branch-and-bound heuristics – refer to Cheng: pg. 371, 374 and the *non-exhaustive versus exhaustive* rationale set forth in claim 5) along individual ones of a plurality of partial critical paths through a data flow graph, wherein the plurality of delays are from a processing element being scheduled to a sink node of the data flow graph (refer to claim 34), and

wherein said performing a first calculation results at least in a first longest-in-duration path calculation of a first longest-in-duration path of the plurality of partial critical paths (refer to claim 34);

determining from among remaining paths of the plurality of partial critical paths,_a new longest-in-duration path and scheduling the new longest-in-duration path from among the remaining paths (refer to claim 34 – Note: algorithm based on source code or DFG cannot stop at first longest-path calculation).

Cheng does not explicitly disclose circuit as an *integrated circuit*, but this limitation has been addressed as obvious in claim 34.

Nor does Cheng explicitly disclose:

performing a *second non-exhaustive or exhaustive calculation a delay through the first longest-in-duration path wherein said performing a second non-exhaustive or exhaustive calculation results in a second longest-in-duration path calculation; and determining whether the second longest-in-duration path calculation confirms the first longest-in-duration path and scheduling, of the first the longest-in-duration path calculation upon a determination that the second longest-in-duration path calculation confirms the first longest-in-duration path calculation.*

Dangelo discloses a chip building method with design phase, implementation and verify phase (see Fig. 16B) including verifications of constraints in a description phase (Fig. 2) where any preliminary timing or path analysis estimation would be verified (see col. 28 line 50 to col. 29 line 26) prior to decision to import the findings into the design, using for critical path delay estimation algorithm, a repeat in re-synthethizing using result from a earlier synthesis phase (e.g. col. 48 line 45 to col. 49 line 14) to obtain a better result or compliancy for the intended design. It would have been obvious for one of ordinary skill in the art to implement the scheduling in Cheng so that the approach of Dangelo is utilized; that is, including performing a second non-

exhaustive of longest-path duration based on the initial longest-in-duration path calculation in order to confirm on result or compliance (using a generated second longest-in-duration calculation) requirement as taught in Dangelo's verification approach thereby performing scheduling or the remaining longest-in-duration algorithm. One would be motivated to do so because reliance on a estimates or results from non-exhaustive phase cannot establish defined and accepted values needed for a design (or compliance thereto) unless repeat of a same algorithmic step would confirm the validity of a initial estimate, lest the reduction endeavor regarding implementation of physical layout (of a ASIC as from Dangelo – see col. 4 lines 46-67) would be defeated by erroneous values.

As per claim 39, Cheng does not explicitly disclose:

calculating performing a third non-exhaustive or exhaustive calculation_with the second, more exhaustive calculation the of a delay through a new longest-in-duration path chosen from among the plurality of partial critical paths excluding the first longest-in-duration_path. But calculating second, then third longest-in-duration based on result of a previous immediate calculation(e.g. second from a first, third from a second) among the remaining paths would be a variant of the "second calculation" limitation having been addressed in claim 38; hence by virtue of the rationale set forth therein, the third non-exhaustive or exhaustive calculation over the second calculation would have been obvious for the same reasons.

As per claim 40, Cheng discloses wherein the exhaustive second non-exhaustive or exhaustive calculation includes branch and bound calculation (*branch-and-bound … more efficient* - pg. 374, top; sec 3.1.1.1 pg. 361; second phase … branch-and-bound – pg. 386 middle).

As per claim 41, Cheng does not explicitly disclose scheduling all shorter-in-duration paths within time established for the first longest-in-duration path or the new longest-in-duration path. But by calculating longest-in-duration, the intent would entail finding a shorter duration path relative to a longer duration path as per the very heuristic by Cheng of finding the longest-in-duration path; hence scheduling all shorter-in-duration would have been inherent or obvious as one would be motivated to find a shorter-in-duration among the longest path in order to optimize the execution time of the circuitry gates and physical layout of the intended circuit via using Cheng's high-level synthesis approach.

As per claim 42, refer to claim 34 or 36.

As per claim 43, Cheng (in view of Dangelo) discloses computer-readable medium comprising stored programming instructions which, in response to execution by a processor of an apparatus, causes the apparatus to perform computer programming the method of claim 38 (refer to claim 38).

17. Claims 44, 46, 50 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al, USPN: 6,745,160 (herein Ashar) further in view of Hammes et al, USPN: 7,155,708 (herein Hammes)

As per claim 44, Ashar discloses a method of fabricating a reconfigurable integrated circuit, comprising:

scheduling an operation of a processing element in a loop (Fig. 1-3; scheduling – col. 7 line 62 to col. 9 line 28);

scheduling one or more operation/signal times following the operation of the processing element in a loop (operations … number of iterations – col. 29 lines 19 to col. 30 line 10);

scheduling an operation of further additional processing elements, wherein the additional processing elements are dependent on the processing element in a loop, and wherein said scheduling an operation schedules the additional processing elements to begin beginning after the one or more operations times to permit communication to the dependent additional processing elements of the status of the additional processing elements (states S 11, S12 col. 29 lines 19 to col. 30 line 10 - Note: inner outer loop and recording of states between inner and outer loops with regard to data flow graph reads on additional processing elements scheduled at the beginning permitting communication to dependent additional elements of the status of said first additional elements – Fig. 12a, b, c)

Ashar does not explicitly disclose operations times as *buffer* times; but the use of gates to receive a read/write or a signal in Ashar's RTL HDL-based development via scheduling and flow graph analysis (see Ashar: col. 8-11) was well-known such as a buffer or a MUX (see Ashar: col. 19 lines 21-50) depicting a logic on operation implicating input/output within control flow analysis. Hammes, as per a control flow graph analysis to derive loop dependencies or node latencies (Hammes: col. 32-33) discloses association between CFG and HDL in addressing pipeline implementation and a buffer gate to receive input from a bit stream (buffer – col. 41 lines 20-44). It would have been obvious for one of ordinary skill in the art to implement the iteration or loop operations in Ashar scheduling so that operational signal uses gates like a buffer to account for the number of signal being passed into the gates per each iteration as set forth in loop scheduling or verification of Ashar, because arrival of signal in a pipeline being implemented via a scheduling endeavor would be facilitated with use of buffering prior to actual processing of the received input by other gates like a MUX as set forth above.

As per claim 46, Ashar discloses scheduling a single iteration of the operation (Fig. 3)

As per claim 50, Ashar (in view of Hammes) discloses a computer-readable medium comprising stored programming instructions (see Ashar: col. 38: readable media) which, in response to execution by a processor of an apparatus, causes the apparatus to perform the method of claim 44 (refer to claim 44)

18. Claim 45 is rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al, USPN: 6,745,160 (herein Ashar) in view of Hammes et al, USPN: 7,155,708, further in view of Dangelo et al, USPN: 5,572,436 (herein Dangelo)

As per claim 45, Ashar does not explicitly wherein said scheduling an operation of a processing element in a loop includes comprises scheduling an estimated number of iterations of the operation. Dangelo disclose taking into account estimate of a particular operation (e.g. power consumption) or parameters (col. 51 line 59 to col. 52 line 44) within the paradigm of repeated algorithm procedures effectuated for optimizing RTL related scheduling or high-level or optimizing HW synthesis. It would have been obvious for one of ordinary skill in the art to implement the loop scheduling and verification by Ashar so that operation of a processing element (a gate collected iterations and represented by a CFG node – see Ashar: Figs. 12) with taking into account of estimated number of iterations as taught in Dangelo, because estimate parameters can be verified via successive attempt to repeat the algorithmic procedures as taught in Dangelo, whereby enabling the final layout of circuit or VHDL components to be proper and compliant to well-known and initial purport of graph based scheduling.

19. Claims 47-48 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al, USPN: 6,745,160 (herein Ashar) in view of Hammes et al, USPN: 7,155,708, further in view of Ruttenberg, USPN: 5,491,823 (herein Ruttemberg)

As per claim 47-48, Ashar does not explicitly disclose comprising providing for notification of delay of all of the additional processing elements during the buffer times in response to the processing element iteratively operating in excess of the scheduled operation and providing for delay of a network schedule manager and of a logic schedule manager upon the processing element iteratively operating in excess of the scheduled operation (Note: providing for delay treated as notification of delay due to excess lifetime of an operation – see USC 112 Rejection).

loop scheduling is disclosed in Ashar as verifying on sufficient threads being present in terms of invariants determination (Fig. 4; col. 9 line 50 to col. 10 line 10) using state transition graph where the analysis uses assumptions prior to proceeding on scheduling iterative cycles (col. 8 line 39 to col. 29) in order to verifying correctness of the assumptions. Analogous to the replication of paths in Ashar (see Ashar: sec 1.2.2.3 col. 5-6), Ruttenberg discloses live range of variable in view of the invariant concept is disclosed in loop scheduler approach and use of lifetime analysis based on which to replicate a schedule (col. 17 line 50 to col. 18, line 16) where value for replicating a loop with adjustment of register resources is based on lifetime of operation instance (col. 1 line 55 to col. 2 line 19). It would have been obvious for one of ordinary skill in the art to utilize the assumptions in Ashar so that lifetime or range of non-invariants affected by the recursive behavior of loops are taken into consideration when addressing delay among iterated operations of gates (e.g. buffer times) particular when a

particular operation element exceeds an assumed value (live range or lifetime as in Ruttenberg) where detection or acknowledging of the event where such assumption is being exceeded (lifetime surpassing a presumed value) would generate a manager type of notification for corrective action to be taken, e.g. via a replication mechanism to readjust register resources as taught in Ruttenberg, obviating memory range conflict.

20. Claims 49, 51, 53 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al, USPN: 6,745,160 (herein Ashar) in view of Hammes et al, USPN: 7,155,708, further in view of APA (Specifications pg. 2-10).

As per claim 49, Ashar does not explicitly disclose an integrated circuit fabricated according to the method of claim 44. Based on the high-level synthesis and RTL/HDL based setting of Ashar, Hammes discloses placement in the context of implementing reconfigurable circuitry (col. 1-2; Fig. 11). APA discloses chip design with partitioning of blocks into sub-blocks of the hardware elements in area of surface substrate(surface planning, recursively into smaller blocks – APA, pg. 9), with routing of interconnections and portioning, placement and routing (APA, Specs pg. 8) using iterative placement until size of chip is reduced (APA, pg. 9-10). It would have been obvious for one of ordinary skill in the art to implement the synthesis and scheduling in Ashar in view of Hammes for the purpose of fabricating a circuit like a ASIC for the same reasons as set forth in claim 2

As per claim 51, Ashar discloses a method, comprising:

providing a first loop having a first processing element; providing an output of the first loop to a second loop having a second processing element; providing an output of the second loop to an input of the first loop (Fig. 2a,b; Fig. 5; Fig. 10, 12);

scheduling an identical number of operations of the first processing element_and the second processing elements (scheduling – col. 7 line 62 to col. 9 line 28; Fig. 2a,b; Fig. 5; Fig. 10, 12 -Note: scheduling via replicated iterative steps of loops reads on identical first and second PE); and

scheduling an operation of further additional processing elements, wherein the additional processing elements are dependent on the first loop and the second loops, and wherein said scheduling an operation schedules the additional processing elements to begin beginning after a operation/signal time to permit communication of the status of the first loop and the second loops to the dependent additional processing_elements (refer to claim 44).

Ashar does not explicitly disclose fabricating a reconfigurable integrated circuit; but this has been rendered obvious as per claim 49, in view of Hammes and APA.

Nor does Ashar explicitly disclose operation/signal time exactly as 'buffer time". But this limitation is deemed obvious and has been addressed in claim 44, in view of Hammes.

As per claim 53, refer to claim 46.

21. Claim 52 is rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al, USPN: 6,745,160 (herein Ashar) in view of Hammes et al, USPN: 7,155,708, and APA; further in view of Dangelo et al, USPN: 5,572,436

As per claim 52, refer to the rationale of 45.

22. Claims 54-55 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al, USPN: 6,745,160 (herein Ashar) in view of Hammes et al, USPN: 7,155,708, and APA; further in view of Ruttenberg, USPN: 5,491,823 (herein Ruttemberg)

As per claims 54-55, refer to the rationale of 47-48 (in view of the USC 112 Rejection)

23. Claims 56, 58, 61 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al, USPN: 6,745,160 (herein Ashar) in view of Cheng et al, "High-Level synthesis: Current Status and Future Prospects", June 1993 pp. 351-400 (herein Cheng) further in view of APA and Hammes et al, USPN: 7,155,708.

As per claim 56, Ashar discloses a method comprising:

providing a loop having control nodes within the loop (refer to claim 44; LOGIC, control –col. 19 lines 21-34);

scheduling the loop by:

scheduling a path through the loop(Fig. 5a; consider a path – col.25 lines 20-57); and

scheduling an operation of processing elements dependent on the loop after a operation/iteration time (operations … number of iterations – col. 29 lines 19 to col. 30 line 10)

Ashar does not explicitly disclose scheduling longest in longest-in-duration path. Cheng discloses longest path analysis within stages for improved high-level synthesis (longest path, maintains data dependencies – pg. 383). Based on the common endeavor in Ashar and Cheng to optimize layout of a circuit design via scheduling of a VHDL application, it would have been obvious for one of ordinary skill in the art to implement analysis of data dependencies regarding datapath and states of invariants (Ashar: col. 25 lines 20 to col. 26 line 15) so that path considering in the loops in the context of verifying data changes per iteration so that longest-duration-path would be included in order to extract the needed reconfiguration of resources based on the delay via confirming of values whereby allocated resources usage can be reallocated to improve the overall layout scheme as endeavored in Cheng's multi-stage scheduling.

Ashar does not explicitly disclose operation/signal time as "buffer time" but this has been addressed in claim 44.

Nor does Ashar disclose method for *fabricating a reconfigurable integrated circuit*. This has been addressed in claim 49.

As per claim 58, refer to claim 46.

As per claim 61, refer to claim 49

24. Claim 57 is rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al,USPN: 6,745,160 (herein Ashar) in view of Cheng et al, "High-Level synthesis: Current Status and Future Prospects", June 1993 pp. 351-400 (herein Cheng) and APA and Hammes et al, USPN: 7,155,708, further in view of Dangelo et al, USPN: 5,572,436

As per claim 57, Ashar discloses scheduling an estimated number of iterations of the operation of the loop (as per the rationale set forth in claim 45), wherein the iterations employing the longest-in-duration path through the loop (refer to claim 56).

25. Claims 59-60 are rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al,USPN: 6,745,160 (herein Ashar) in view of Cheng et al, "High-Level synthesis: Current Status and Future Prospects", June 1993 pp. 351-400 (herein Cheng) and APA and Hammes et al, USPN: 7,155,708, further in view of Ruttenberg, USPN: 5,491,823 (herein Ruttemberg)

As per claims 59-60 (in view of the USC 112 Rejection), refer to claims 47-48

26. Claim 62 is rejected under 35 U.S.C. 103(a) as being unpatentable over Ashar et al, USPN: 6,745,160 (herein Ashar) in view of Cheng et al, "High-Level synthesis: Current Status and Future Prospects", June 1993 pp. 351-400 (herein Cheng) further in view Hammes et al, USPN: 7,155,708.

As per claim 62, refer to claim 56 for Ashar disclosing computer readable medium having:

instructions to provide a loop having control nodes within the loop;

instructions to a longest-in-duration path through the loop (Cheng); and

instructions to schedule an operation of processing elements dependent on the loop after a buffer time (Hammes), wherein communication of a status of the loop to the processing elements is permitted.

Allowable Subject Matter

27. Claims 13-25 are objected to as being dependent upon a rejected base claim, but would be allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims.

The subject matter deemed allowable contains:

(a) identifying seed basic blocks by identifying candidate seed basic blocks among the identified basic blocks of the of control flow graphs, and comparing candidate seed basic blocks from control flow graphs of separate program operations.

identifying ones of the_basic blocks that lie inside a loop, including ones of: a single nested level loop with only one basic block; a single nested level loop with more than one basic block; and a multi-level nested loop, a single nested level loop with more than one basic block; and a multi-level nested loop

(b) identifying basic blocks of control flow graphs of separate program operations under like control; determining a count of individual operation types in a basic block, including examining edges in a data flow graph of candidate seed basic blocks of control flow graphs from

the separate programming operations; classifying edges based on source and destination node operation type;

(c) wherein said examining edges includes eliminating edges of one data flow graph having a source-operation-to-destination-operation not found in another data flow graph having edges under examination; implementing the eliminated edges in a circuit other than in an application specific integrated; implementing the eliminated edges with in one or_more look up tables.

Conclusion

28. Any inquiry concerning this communication or earlier communications from the examiner should be directed to Tuan A Vu whose telephone number is (571) 272-3735. The examiner can normally be reached on 8AM-4:30PM/Mon-Fri.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Lewis Bullock can be reached on (571)272-3759.

The fax phone number for the organization where this application or proceeding is assigned is (571) 273-3735 (for non-official correspondence - please consult Examiner before using) or 571-273-8300 (for official correspondence) or redirected to customer service at 571-272-3609.

Any inquiry of a general nature or relating to the status of this application should be directed to the TC 2100 Group receptionist: 571-272-2100.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

/Tuan A Vu/

Primary Examiner, Art Unit 2193

July 28, 2010